MODELING AND CONTROL OF THE AGV SYSTEM IN AN AUTOMATED CONTAINER TERMINAL

Qin Li, Jan Tijmen Udding, and Alexander Yu. Pogromsky
Mechanical Engineering Department
Eindhoven University of Technology
Eindhoven, The Netherlands
email: {q.li, j.t.udding, a.pogromsky}@tue.nl

ABSTRACT
We study a modeling and traffic control problem of the automated guided vehicle (AGV) system in a container terminal. The workspace of the vehicles (straddle carriers) is modeled as a road network composed of lanes, crosses and depots. The container transport task for each vehicle is abstracted as a finite sequence of zones to be traversed by a vehicle, starting at the initial zone of the vehicle and ending with some depot. A set of traffic rules is proposed to ensure the completion of all jobs with the absence of vehicle deadlocks and collisions. The traffic rules are very simple and real-time implementable. Moreover, these rules can be realized almost distributedly requiring little intervention from a central controller.

KEY WORDS
Modeling, traffic control, deadlock avoidance, distributed algorithm

1 Introduction
In a container terminal (see Figure 1), the vessels are discharged and loaded by quay cranes (QC) at the sea side. The containers that are discharged are put onto a prime mover, or put on the ground, to be picked up by a straddle carrier, for horizontal transport to the Yard side. Here yard stackers (YS) must be used to put the containers into their final storage location in the case of using prime movers. Straddle carriers can be used in combination with YSs as well, although they can transport the containers directly to their final destinations. The other way around, the containers to be loaded onto vessels are collected from the yard by the YSs or straddle carriers and transported to the designated QCs by either straddle carriers or prime movers. Since straddle carriers are capable of picking up and dropping off containers by themselves, they have the advantage of not requiring synchronization with a QC or an YS.

As international trade steps up, automated container terminals (ACTs), equipped with automated container transshipment systems, are appealing in meeting the increasing demand for higher operational efficiency and lower costs than what traditional terminals can achieve. Moreover, automated terminals tend to exhibit a smaller variability. Three issues are the main concerns for ensuring a desirable performance of an ACT. The first one is the dispatching problem which is about where and when a vehicle (straddle carrier or prime mover) should go for a container loading or discharging task. The second is the vehicle routing problem which is on the path planning for each dispatched vehicle. The third is called the traffic control problem, i.e., how to resolve motion conflicts in the vehicle system. In this paper, we address the last problem with the main purpose to ensure the absence of deadlocks and collisions of the automated guided vehicles, called AGVs hereafter.

The study on deadlock problems was originated in the development of operating systems [7, 8] and database systems [20] in computer science. In general, three strategies have been employed to address deadlocks: deadlock prevention, deadlock avoidance and deadlock detection and resolution [20]. The goal of deadlock prevention is to design rules such that any deadlock is pre-excluded before any process sets off. In a deadlock avoidance strategy, online control is adopted such that a resource is allocated to a process only if a deadlock will not be introduced. Lastly, a deadlock detection and resolution strategy allows deadlock to happen. When a deadlock is detected, certain online rules are used to resolve the deadlock and recover the system. During the past two decades, deadlock problems have been attracting intensive research efforts in the design of automated manufacturing systems (AMSs) in which
AGVs are used in the material handling systems (MHSs). In particular, [16, 15, 17, 6] are among many other works which adopt Petri net models to design deadlock avoidance schemes on the control of transition firings. Using a graph theoretic approach, [19] presented a deadlock avoidance algorithm which consists of a quasi two-step resource allocation policy and a cyclic deadlock detection procedure. The authors of [2] and [3] proposed deadlock avoidance policies based on the graphic characterization of the deadlocks and the so called restricted deadlocks. In [4], a variation of the Banker’s algorithm was provided for deadlock prevention. The authors of [1] gave a simple cyclic deadlock prediction algorithm which has very small time complexity. [14] offers a logic-algebraic approach for deadlock prevention by solving a constraint satisfaction problem. On the other hand, some authors considered the conflict-free shortest-time routing which excludes the possibility of the occurrence of deadlocks in the route selection phase [9, 10, 11]. Recent years have witnessed increasing research attention on solving deadlock problems for the AGV systems at ACTs. [12] presented several procedures to detect and recover from deadlocks caused by the interaction between the AGV system and other handling equipments. [5] renders a path reservation method to prevent in advance the deadlocks and collisions of AGVs. The authors of [18] suggested a traffic control approach based on the concept of “semaphore” borrowed from computer science. In addition, inspired by [1], the work [21] showed a deadlock detection and avoidance strategy involving some on-line cyclic deadlock checking procedures.

In this work, the work area of the vehicles in a terminal is modeled as a road network which consists of lanes, crosses and depots. To ease the conflict resolution, every lane is regarded unidirectional and composed of a series of zones, where each zone is large enough to accommodate at least one vehicle. The container transshipment task for each vehicle is abstracted as a finite sequence of zones to be traversed by the vehicle, starting at the initial zone of the vehicle and ending with some depot. In this way of modeling, the traffic control problem under consideration falls into the category of the classic traffic control problems for zone-control AGV systems. As pointed out in [21], the relatively large numbers of zones and vehicles at an ACT defy most existing deadlock avoidance approaches which rely on either matrix operations or complex search and thus are highly expensive. Motivated by [1] and [21], in this paper we propose a set of real-time implementable traffic rules that involves a cyclic deadlock detection procedure. Based on the traffic rules, each vehicle makes a “go ahead” or “stop” decision after probing if moving forward can lead to a collision or a cyclic deadlock. The traffic rules are very simple and can be implemented almost distributedly, i.e., they can be realized mostly by local inter-vehicle sensing and communication with a slight intervention from a central controller needed to coordinate the vehicles’ at-cross behavior. Unlike in [1] and [21] where the vehicles may be bogged in restricted deadlocks [3] (namely the multi-cycle deadlocks [21]), we will prove that by our proposed method all vehicles can complete their jobs free of deadlocks and inter-vehicle collisions. One will find that our modeling and control schemes can also be applied to many other applications that share similar zone-control essentials.

The rest of the paper is organized as follows: Section 2 gives the description of the zone-control model of the work area in a container terminal. Then the problem to be addressed in the paper is formalized in Section 3. The traffic rules are elaborated in Section 4, where the formal proof for the deadlock and collision avoidance is also provided. Finally, some concluding remarks summarizing the features of the proposed traffic control strategy and some comments on the direction of future work are given in Section 6.

2 Zone-Control Modeling

In this section, we introduce some basic notions in our zone-control modeling. Throughout the paper, for any $n_1, n_2 \in \mathbb{Z}$, we use the notation $n_1, n_2$, to denote the set \( \{ n_1, n_1 + 1, \ldots, n_2 \} \) if \( n_2 \geq n_1 \), or \( \emptyset \) if \( n_2 < n_1 \).

2.1 Elements in the zone-based road network

The workspace is modeled as a road network composed of lanes, crosses and depots.

Lane: A lane is a road segment on which a vehicle can move with predefined direction. The direction that vehicles are allowed to move along on a lane is called the direction of the lane. The set of lanes in the system are denoted by \( \mathcal{L} = \{ l_1, l_2, \ldots, l_M \} \).

Zone: Lane \( l_i \in \mathcal{L} \) is divided into \( m_i \) zones along its direction, each of which has a dimension that can accommodate one vehicle. On each lane, the \( k \)th zone with respect to the direction of the lane is called the \( k \)th zone of the lane. We use \( c^k_{l_i} \) to denote the \( k \)th zone of lane \( l_i \), and \( C_{l_i} = \#C_{l_i} = m_i \), to denote the set of zones of lane \( l_i \). Particularly, we call the first zone and the last zone of a lane the starting zone (SZ) and ending zone (EZ) of the lane respectively.

Each depot is modeled also as a zone, sometimes called a depot zone, which can accommodate any number of vehicles. We emphasize that a depot is a zone that is not on any lane. Each depot is affiliated with at least one in-lane and one out-lane. An in-lane (resp. out-lane) of a depot is a lane with the direction that allows a vehicle moving towards (resp. outwards from) the depot.

Cross: A cross is a junction area connecting multiple lanes. A lane with the direction that allows a vehicle moving towards (resp. outwards from) a cross is called an in-lane (resp. out-lane) of the cross. The EZ of any in-lane of a cross is named an at-cross zone of the cross. The zones which are not at-cross zones are called off-cross zones. Note that any depot zone must be an off-cross zone as it is not on any lane.
Neighboring lanes and neighboring zones: Any out-lane of a cross (depot) is a neighboring lane of any in-lane of the cross (depot). The neighboring zones of each depot are the SZs of all its out-lanes. The neighboring zone of a non-EZ zone in-lane $c_k^i$, $k \in 1, m_i - 1$ on any lane $l_i \in L$ is $c_{k+1}^i$, i.e. the next zone w.r.t. the direction of $l_i$. The neighboring zone of the EZ of any in-lane of a depot is the depot. The neighboring zones of the EZ of an in-lane of a cross (i.e. an at-cross zone) include the SZs of all the out-lanes of the cross. We use $\mathcal{T}_c$ to denote the set of neighboring zones of zone $c$.

Quay crane and yard stacker: We model the container buffer area of each QC or YS as a cross, called station cross, which has at least one in-lane and one out-lane.

Figure 2 illustrates a layout of the road network for the workspace of a simplified container terminal (with two QCs, one YS and one depot), where vehicles are represented by circles with indices.

Regarding the layout of the zone-control model, we make the following assumption.

**Assumption 1** (a) Every lane in $\mathcal{L}$, except the in- and out-lanes of each depot, has at least two zones; (b) Each lane is an in-lane of either a cross or a depot, it is also an out-lane of a cross or a depot; (c) Any in-lane of a cross has at least one neighboring lane.

**Remark 1** By Assumption 1, it is not difficult to see that every zone has at least one neighboring zone.

2.2 Vehicle events and states

Each vehicle has two possible states: being “in $c_1$” and “moving from $c_1$ to $c_2$”, where $c_1$ is any zone and $c_2 \in \mathcal{T}_{c_1}$.

When the body of a vehicle has completely entered zone $c_1$, and the position reference point (PRP) of the vehicle is detected to be in the arrival region (see Case (a) in Figure 3), the vehicle is said to arrive at the zone and changes its state to be in $c_1$. When a vehicle is heading from $c_1$ to $c_2$, and its PRP has moved into the departure region of the zone (see Case (b) in Figure 3), the vehicle is said to intend to leave $c_1$. At this moment, for the sake of collision and deadlock avoidance, the vehicle is required to refer to some traffic rules (to be specified in Section 4) to make a “go ahead” or “stop” decision. When the vehicle is permitted to go ahead, the vehicle is said about to leave zone $c_1$ and changes its state to be moving from $c_1$ to $c_2$; otherwise it has to stop and wait until it is allowed to go ahead by the traffic rules.

![Figure 3: vehicle’s arrival and departure at a zone](image)

We call the “arriving at”, “intending to leave” and “about to leave” the vehicle events, happening sequentially when a vehicle passes a zone, where the first two are position-based events and the third is a decision-based event. Note that a vehicle switches its state only when it arrives at or is about to leave some zone, and must be in some zone when it intends to leave the zone.

A vehicle is said to be an at-cross vehicle if it is in some at-cross zone. A vehicle is said to be passing cross $r$ if the vehicle is moving from the EZ of some in-lane of $r$ to the SZ of some out-lane of $r$. We sometimes say zone $c$ contains vehicle $v$ if $v$ is in $c$.

The following assumption is made for the purpose of inter-vehicle collision avoidance.

**Assumption 2** (a) Let $v_1$ and $v_2$ be any two vehicles, then $v_1$ does not collide with $v_2$ if either of the following conditions is true: (i) $v_1$ is in $c_1$ and $v_2$ is in $c_2$; (ii) $v_3$ ($v_2$) is in $c_1$ and $v_2$ ($v_1$) is moving from $c_2$ to $c_3$; (iii) $v_1$ is moving from $c_1$ to $c_2$ and $v_3$ is moving from $c_3$ to $c_4$, where the non-depot zones in each of the three cases are distinct.

(b) For any depot, there is at most one vehicle which can be moving from the depot to the SZ of any out-lane of the depot at any time.

Note that Assumption 2 suggests some vehicle motion coordinations at the depots, e.g. two vehicles in a same depot should be guaranteed not colliding with each other and not allowed simultaneously moving to a common neighboring zone of the depot. The realization of these coordinations will not be discussed in this work.
2.3 Zone states

Each zone can have two states: being “occupied” and “available”. Each depot zone is always available. Non-depot zone is said to be occupied by a vehicle if the vehicle is either in; (b) moving from some zone to ; or (c) moving from to some zone. Non-depot zone is available if it is not occupied (by any vehicle).

To ease presentation, we sometimes use colors to represent different status of the zones. Specifically,

(1) Zone is white if is available;

(2) Zone is black if is occupied by only one vehicle which is either in or moving from some zone to ;

(3) Zone is grey if is occupied by only one vehicle which is moving from to some zone.

We will prove that by our proposed traffic rules, at any moment, each non-depot zone can be occupied by at most one vehicle and hence can have either one of the above colors. (See Corollary 1 in Section 4.)

3 Problem Formulation

In this section, we describe the operation scenario of a team of vehicles, represented by the set , in a container terminal, from which we extract the main problem studied in this paper.

First we impose an assumption on the initial status of the vehicles.

Assumption 3 At initial time , each vehicle is in some zone (called the starting zone of the vehicle), and each non-depot zone is occupied by at most one vehicle.

It follows directly from Assumption 3 that each zone is initially either black or white.

The operation scenario of vehicles can be briefly described as follows: After the initial time , each operational vehicle may visit one or more QC or YS. After a vehicle arrives at the EZ of an in-lane of a QC or a YS, it needs to pass the station cross (representing the container unloading or loading processes) and move to the SZ of the station, where a new container transport task will be assigned. It is required that each vehicle park in a depot eventually.

Based on the operation description, each vehicle is assigned a job for container transport as defined below.

Definition 1 The job of vehicle is a finite sequence of zones , where for all , is the starting zone of the vehicle, and is some depot zone.

Now let us introduce some instrumental notions in this work before formally state the main problem under our investigation.

Definition 2 (Current and next zones of a vehicle) If a vehicle is in some zone, then the zone is said to be the current zone of the vehicle. The next zone of a vehicle at time is the zone the vehicle should arrive first after it according to the given job. The previous zone of a vehicle at time is the last zone the vehicle was about to leave before it.

Definition 3 (Next zone(s) of a zone) The set of next zones of an off-cross zone is identical to the set of the neighboring zones of the zone. Let be any at-cross zone. If there is a unique vehicle in or moving from to the next zone of at time , then the next zone of at is defined to be the next zone of at . Otherwise, does not have a next zone at .

Remark 2 The current and next zones of a vehicle vary with time, and each vehicle can have at most one current or next zone at any time. The next zone(s) of any off-cross zone is time-invariant, but the next zone of an at-cross zone may change with time.

Definition 4 (Operation graph and cycle) The operation graph is a time-dependent graph with the node set and arc set . A directed cycle (or simply cycle) in is a sequence of nodes such that and are all distinct; and is a cycle in . We say that a node in is white (black or grey) if and only if the corresponding zone is in that color. A black cycle in the graph is a cycle in with all nodes black.

As one will see, the guarantee of no black cycle in is vital for our traffic control scheme. The following assumption excludes the existence of the black cycles at initial time.

Assumption 4 There is no black cycle in .

Now we are in a position to formulate the main problem concerned in this paper:

(P1): Under Assumptions 1-4, design a set of traffic control rules so that any set of jobs can be completed, i.e., each vehicle is successively in each zone of its job, and inter-vehicle collisions are avoided.

4 Traffic Control Strategy

As mentioned in Section 2, when a vehicle intends to leave some zone, the vehicle is required to refer to some traffic rules to make a “go ahead” or “stop” decision. In this section we specify these traffic rules. We differentiate the cases where a vehicle intends to leave an off-cross zone and an at-cross zone. It will be proved that the problem P1 can be solved by our proposed traffic rules.
4.1 Traffic rules

Firstly, for the purpose of the collision avoidance we should not allow a vehicle to head for its next zone if the next zone is occupied by another vehicle. Indeed this minimum requirement becomes our traffic rule for off-cross zones. Specifically, the following Rule 1 applies when a vehicle intends to leave an off-cross zone.

**Rule 1: traffic rule for off-cross zones**

Suppose a vehicle intends to leave an off-cross zone \( c \) with \( c^* \) the next zone of the vehicle. It waits until \( c^* \) is available, then change its state to be “moving from \( c \) to \( c^* \”).

Now, note that applying Rule 1 to every zone may result in a failure in solving the problem P1. In particular, it cannot guarantee the absence of such a scenario where each of a set of vehicles is in a non-depot zone and the set of (black) zones that contain these vehicles form a black cycle in the graph \( G_o \). In this case, the vehicles are said to be in a cyclic deadlock and none of them can leave its current zone according to Rule 1. This observation motivates us to put forward a different traffic rule for at-cross zones, which involves a procedure for the cyclic deadlock prediction (CDP).

The following Rule 2 applies when a vehicle intends to leave an at-cross zone.

**Rule 2: traffic rule for at-cross zones**

Suppose a vehicle intends to leave an at-cross zone \( c \) with \( c^* \) the next zone of the vehicle. It waits until (a) \( c^* \) is available and (b) to change its state to be “moving from \( c \) to \( c^* \)” will not lead to a black cycle in \( G_o \) containing \( c^* \), then it changes its state to be “moving from \( c \) to \( c^* \”).

An algorithm used to check the condition (b) in Rule 2 is called a CDP procedure. Note that if the global information of the AGV system is known to each vehicle, the CDP procedure can be performed by an individual vehicle. Otherwise it has to be accomplished in a distributed and cooperative manner. We postpone the discussion on the latter case to Subsection 4.3, where a distributed CDP procedure is given.

In this paper, it is inhibited that two or more vehicles are executing the CDP procedure simultaneously. This can be practically realized by letting each vehicle ask a central computer (controller) for a unique global “token” (called the CDP token hereafter) to trigger the execution. On the other hand, it is necessary to avoid that some at-cross vehicle waits endlessly for the token to start executing the CDP procedure. In summary, we impose the following assumption on performing the CDPs.

**Assumption 5** At most one at-cross vehicle is executing the CDP procedure and can change its state at any time. If some vehicle intends to leave an at-cross zone at time \( t_1 \), then there is some time \( t_2 > t_1 \) such that the vehicle is offered the CDP token and starts executing the CDP procedure at \( t_2 \).

**Remark 3** One interesting topic here is how to elaborately assign the CDP token to a queue of at-cross vehicles that are requesting it at the same time in order to, in some sense, maximize the performances of the AGV system (i.e. throughput, timeliness etc.). Careful exploration of this issue however is beyond our efforts for this work. In our current simulation studies, we feed the at-cross vehicles with the token by a simple heuristic scheduling which will be briefly mentioned in Section 5.

4.2 Deadlock and collision avoidance

Now we justify that problem P1 can be solved by the proposed traffic rules. Indeed, this claim follows directly from Theorems 1 and 2 below. Sketched proofs are provided here for want of space.

**Theorem 1** Following the proposed traffic rules, each non-depot zone can be occupied by at most one vehicle at any time, and no inter-vehicle collision can occur.

**Proof:** By contradiction, define the time instant \( t^* \) as \( t^* := \inf \{ t : \) there is some non-depot zone occupied by more than one vehicles at \( t \} \). It can be shown that there exists some vehicle that changes its state from “in \( c_1 \)” to “moving from \( c_1 \) to \( c_2 \)” at \( t^* \) when \( c_2 \) (a non-depot zone) is occupied by another vehicle. This however is prohibited by either Rules 1 or Rule 2.

For the second part of the theorem, just note that if two vehicles \( v_1 \) and \( v_2 \) are not occupying a same non-depot zone they must fall into either of the three scenarios depicted in Assumption 2(a).

Theorem 1 has the following immediate corollaries.

**Corollary 1** Each zone is either white, black or grey at any time.

**Corollary 2** Each non-depot zone can have at most one next zone.

**Theorem 2** Following the proposed traffic rules, all the jobs are completed in finite time.

The remainder of this subsection will be devoted to a proof of Theorem 2. Firstly, we relate state changes of the vehicles based on Rules 1 and 2 to a set of zone-coloring procedures, which can be readily verified.

**Arrival Procedure:** When a vehicle arrives at zone \( c_2 \), it colors its previous zone \( c_1 \) white. Additionally, if \( c_2 \) is an at-cross zone, then create an arc in \( G_o \) tailed at \( c_2 \) and pointing to the next zone of \( c_2 \); if \( c_1 \) is an at-cross zone, then destroy the existing arc \((c_1, c_2)\) in \( G_o \).

Rule 1 can be stated as the following off-cross zone departure procedure. Three cases are differentiated due to the constant availability of the depots.
Off-Cross Zone Departure Procedure: Suppose a vehicle \( v \) intends to leave an off-cross zone \( c \) with \( c^* \) the next zone of \( v \). (a) If \( c \) is neither a depot nor the EZ of an in-lane of a depot, then \( v \) waits until \( c^* \) is white and then colors \( c \) grey and \( c^* \) black. (b) If \( c \) is a depot zone, then \( v \) waits until \( c^* \) is white and then colors \( c^* \) black. (c) If \( c \) is the EZ of some in-lane of a depot, then the vehicle colors \( c \) grey.

Rule 2 can be rephrased using the at-cross zone departure procedure.

At-Cross Zone Departure Procedure: Suppose a vehicle \( v \) intends to leave an at-cross zone \( c \) with \( c^* \) the next zone of \( v \). \( v \) waits until (a) \( c^* \) is white and (b) to color \( c \) grey and \( c^* \) black will not lead to a black cycle containing \( c^* \) in \( G_o \), then colors \( c \) grey and \( c^* \) black.

Clearly, the colors of zones and the neighboring topology of the graph \( G_o \) can be changed only by the above arrival and departure procedures.

Lemma 1 The neighboring topology of the graph \( G_o \) can be changed only by the arrival procedures.

Lemma 2 Any arrival procedure or off-cross zone departure procedure cannot introduce black cycle in \( G_o \).

Proof: The part for the arrival procedure follows from two facts: (a) the arrival of a vehicle at an off-cross zone only results in a grey-to-white change of a node and a possible arc deletion in \( G_o \); (b) the arrival of a vehicle at an at-cross zone \( c^* \) creates an arc with the only incoming arc of \( c^* \) in \( G_o \) is set at a zone on the same lane with \( c^* \) which turns white. (By Assumption 1(a) the lane containing \( c^* \) must have at least two zones.)

To see the part for the off-cross zone departure procedure, just note that in an off-cross zone departure procedure a non-depot zone maybe colored black, but the only incoming arc of it in \( G_o \) is set at the off-cross zone which either turns grey (if the zone is not a depot) or keeps white (if the zone is a depot).

Next we prove that there can never be a black cycle in the graph \( G_o \).

Lemma 3 If \( G_o(t_0) \) has no black cycle, then \( G_o(t) \) does not contain a black cycle for all \( t \geq t_0 \).

Proof: Due to the condition (b) of Rule 2.

Theorem 2 is a direct consequence of the following Lemma 4.

Lemma 4 At least one vehicle can arrive at its next zone before all the jobs are finished.

Proof: By contradiction, suppose the claim of the lemma is not true. Then there exists some finite time \( t^* \) such that each zone is either black (if there is a vehicle in it) or white (if it is not occupied) at \( t^* \) and does not change its color for all \( t \geq t^* \), and \( G_o(t) = G_o(t^*) \) for all \( t \geq t_0 \). By Lemma 3, it follows that there exists in \( G_o(t^*) \) a path containing only black zones originated from \( c \) and ended with zone \( c_1 \) whose next zone \( c_2 \) is white. By Assumption 5 and Rule 2, the only reason for the vehicle in \( c_1 \) cannot be granted permission to proceed for all \( t \geq t^* \) is that \( c_1 \) is an at-cross zone and coloring \( c_2 \) black and \( c_1 \) grey leads to a black cycle in \( G_o(t^*) \) containing \( c_2 \). In addition, the black cycle would be the only one containing \( c_2 \). Thus there must be some black at-cross zone \( c_3 \neq c_1 \) at \( t^* \) involved in the black cycle whose next zone is \( c_2 \). However, coloring \( c_2 \) black and \( c_3 \) grey will not incur black cycle containing \( c_2 \) in \( G_o(t^*) \). Therefore, the vehicle in \( c_3 \) will be allowed to proceed at some time \( t \geq t^* \) according to Rule 2.

4.3 Distributed cyclic deadlock prediction algorithm

Recall that in Rule 2, we need to check if the state change of an at-cross vehicle will induce a black cycle. In practice, if the global information of the AGV system is inaccessible to each vehicle then we need a distributed procedure to realize the traffic rule. The following Rule 2’ can be used to substitute Rule 2 for this purpose.

Rule 2’:

Suppose vehicle \( v \) intends to leave an at-cross zone \( c \) of the cross \( r \) with \( c^* \) the next zone of \( v \). The vehicle repeats the following three steps until it is permitted to change its state and proceed (in Step 3).

Step 1: Asks for the CDP token.

Step 2: When granted the token, checks whether \( c^* \) is available and there is no vehicle is passing the cross \( r \). If this is not the case, releases the token and stops; otherwise goes to Step 3.

Step 3: Executes Algorithm 1 below. If the algorithm outputs \( FIND CD = 0 \), then the vehicle changes its state to be “moving from \( c \) to \( c^* \)”, releases the token, and proceeds; otherwise (if \( FIND CD = 1 \)) the vehicle releases the token, and stops.

Algorithm 1 (distributed CDP procedure):

1. Assume \( v \) is moving from \( c \) to \( c^* \)
2. \( \Gamma \leftarrow \{c^*\}, \Lambda \leftarrow \emptyset \)
3. \( z \leftarrow \) the next zone of \( c^* \)
4. \( FIND CD \leftarrow 0, IL \leftarrow 0 \)
5. while (there is some vehicle \( \hat{v} \notin \Lambda \) in \( z \) or is moving from some zone to \( z \) and \( FIND CD = 0 \) and \( IL = 0 \))
6. if \( z \) is some depot, then \( IL \leftarrow 1 \)
7. else if \( z \notin \Gamma \), then
8. \( \Gamma \leftarrow \Gamma \cup \{z\}, \Lambda \leftarrow \Lambda \cup \{\hat{v}\} \)
9. if \( z \) is an off-cross zone, then
10. \( z \leftarrow \) the next zone of \( z \)
11. else \( z \leftarrow \) the next zone of \( \hat{v} \)
12. end
13. else \( FIND CD \leftarrow 1 \)
14. end
15. end
16. end
Remark 4 Note that the implementation of Algorithm 1 requires only inter-vehicle communication. The detailed communication protocol that realizes the algorithm is beyond the scope of this paper.

Now we show that the time complexity of Algorithm 1 is linearly dependent on the number of vehicles, which meets the real-time implementation requirement.

Proposition 1 The running time of Algorithm 1 is $O(N)$.

Proof: It suffices to show that the algorithm ends after at most $N + 1$ iterations of the while loop. In fact, if this was not the case, then $N + 1$ distinct vehicles would have been included in $\Lambda$ before the end of the algorithm.

The efficiency of Rule 2’ can be validated by the following proposition, where $t_s$ and $t_f$ denote the starting and finishing time of Algorithm 1 respectively. The proof are omitted for want of space.

Proposition 2 If $c^*$ is available and there is no vehicle passing $r$ at $t_s$, then the outcome of Algorithm 1 is $FIND_C D = 1$ at $t_f$ if and only if coloring $c$ grey and $c^*$ black at $t_f$ leads to a black cycle in $G_s(t_f)$ containing $c^*$.

Remark 5 Due to Proposition 2, in Step 3 of Rule 2’ we say that a potential cyclic deadlock is detected if the output of Algorithm 1 is $FIND_C D = 1$.

5 Simulation Studies

In this section, we briefly introduce our computer simulations on applying the proposed modeling and control methods to the AGV system in an automated container terminal. Details of the simulation techniques and results and the system performance evaluations will be presented in our follow-up papers. The first step of the simulation is the layout design of the road network (composed of lanes, crosses and depots) for the practical scenario of the application. The main transportation workspace of the vehicles, in which the road network should fill, is a long and narrow strip-like transportation area (2000m×30~40m). Each vehicle is of rectangular shape with a dimension 10m×5m, which means the width of every zone has to be greater than 5 meters. Taking into account the braking distance of the vehicles, the safe size of each zone is fixed to be 22m×7m. To determine the size of each cross, we also need to consider the turning radius of the vehicles. Based on these fundamental specifications, several types of layouts have been put forward and will be compared in terms of performance measures. In the simulations, all vehicles start with a depot and each of them is required to complete a same number of randomly assigned tasks (a task means a QC-QC, QC-YS, YS-YS, or YS-QC journey), and then back to the depot. The route for each task of each vehicle (a sub-sequence of the job in Definition 1) is selected using on a shortest distance algorithm. For the CDP token assignment, we adopt a starvation based scheduling, that is, among all the vehicles that are waiting for the token, the one that has the longest waiting time gets it. We have tested our Rules 1 and 2(2’) with up to 200 vehicles, each of which must complete up to 40 tasks before going back to the depot. In all simulations, no deadlock was found and all vehicles finished their job successfully. In particular, in a simulation with 200 vehicles, 40 tasks for each vehicle, 226 potential cyclic deadlocks were detected by Algorithm 1, but all of them were prevented from turning into real deadlocks due to Rule 2(2’).

6 Concluding Remarks

The most appealing advantage of the proposed modeling and traffic control schemes is that it renders a large range of freedom for the vehicle routing and scheduling aimed at the performance optimization. In detail,

(1) The traffic control is decoupled from the routing of the vehicles. Once the initial states of vehicles satisfy Assumptions 3 and 4, any feasible routes (ending with a depot), i.e. jobs, can be chosen.

(2) The traffic control allows on-line changes of the job of any vehicle. Indeed each agent can change part or the whole of its current remaining job as long as the modified job is still feasible, i.e. satisfies Definition 1.

(3) The order in which the at-cross vehicles get the token for the cyclic deadlock prediction (CDP) is arbitrary provided Assumption 5 is satisfied.

These features admit our traffic rules to be imposed upon any routing algorithm to ensure safety and deadlock free. Also note that (2) is useful for the system to deal with unexpected disturbance. When some obstacle or vehicle breakdown appears on some lane, the (other) vehicles can be re-routed to bypass it.

From an implementation point of view, the decision-making process based on the proposed traffic rules has the following merits:

(1) It has a small time complexity and thus is suitable for real-time executions.

(2) It can be implemented in an almost distributed mode, demanding moderate data communication and slight intervention from a central controller.

Currently, we are building a software platform that will be used to evaluate the overall performance of the AGV-based container transshipment system in a simulated environment of a container terminal after incorporating more sophisticated strategies for job dispatching and vehicle routing and scheduling.

Some future research efforts will be targeted at removing the requirement that each vehicle has to grab the CDP token for the decision-making process based on Rule
2 (Rule 2'). This work will make the traffic control completely distributed and improve the performance of the transportation system by allowing concurrent executions of the cyclic deadlock prediction procedures.

Acknowledgements

The research leading to these results has received funding from the European Community’s Seventh Framework Programme (FP7/2007-2013) under grant agreement no. INFSO-ICT-223844.

References


